# Improving Generalization in Meta-Learning via Meta-Gradient Augmentation

**Ren Wang**[1,2] , **Haoliang Sun**[1*] , **Yuxiu Lin**[3] , **Xinxin Zhang**[1] , **Yilong Yin**[1]
[1]School of Software, Shandong University
[2]Department of Computer Science and Technology, Tsinghua University
[3]School of Computing and Artificial Intelligence, Shandong University of Finance and Economics
{xxlifelover, linxx0109}@gmail.com, {haolsun, xxzhangsdu, ylyin}@sdu.edu.cn

## Abstract

Meta-learning methods typically follow a two-loop framework, where each loop potentially suffers from notorious overfitting, hindering rapid adaptation and generalization to new tasks. Existing methods address this by enhancing the mutual-exclusivity or diversity of training samples, but these data manipulation strategies are data-dependent and insufficiently flexible. This work proposes a data-independent **M**eta-**G**radient **Aug**mentation (**MGAug**) method from the perspective of gradient regularization. The key idea is first to break the rote memories by network pruning to address memorization overfitting in the inner loop, then use the gradients of pruned sub-networks to augment meta-gradients, alleviating overfitting in the outer loop. Specifically, we explore three pruning strategies, including *random width pruning*, *random parameter pruning*, and a newly proposed *catfish pruning* that measures a Meta-Memorization Carrying Amount (MMCA) score for each parameter and prunes high-score ones to break rote memories. The proposed MGAug is theoretically guaranteed by the generalization bound from the PAC-Bayes framework. Extensive experiments on multiple few-shot learning benchmarks validate MGAug's effectiveness and significant improvement over various meta-baselines.

## 1 Introduction

Meta-learning aims to rapidly adapt to unseen tasks by observing learning processes over a wide range of tasks [Hospedales *et al.*, 2021], and has been applied to various scenarios, including few-shot learning [Jamal and Qi, 2019; Xu *et al.*, 2021], continual learning [Martins *et al.*, 2023], transfer learning [Zhang *et al.*, 2023], etc. Current prevalent meta-learning methods follow a unified two-loop framework [Goldblum *et al.*, 2020]. In the outer loop, a meta-learner explores meta-knowledge from numerous tasks. Based on the meta-knowledge, base learners in the inner loop are expected to quickly fine-tune and adapt to new tasks.

As indicated in [Guiroy *et al.*, 2019; Yao *et al.*, 2021], the two-loop framework potentially suffer from meta-overfitting in two aspects: *memorization overfitting* and *learner overfitting*. Memorization overfitting [Rajendran *et al.*, 2020] means that the base learner handles tasks merely based on meta-knowledge rather than task-specific fine-tuning in the inner loop. In this way, meta-knowledge degenerates into rote memorization, which hinders rapid adaptation to new tasks. Learner overfitting [Yin *et al.*, 2020] occurs when the meta-learner overfits to insufficient training tasks, typically manifested by a meta-learner that can adapt quickly but still fails on new tasks. Those two forms of overfitting greatly degrade the generalization and robustness of meta-learning methods.

Data manipulation is a simple yet efficient strategy to tackle these two overfitting issues [Ni *et al.*, 2021; Yao *et al.*, 2021], including constructing mutually-exclusive tasks and conducting task-level augmentation. The former works on addressing memorization overfitting, where training tasks are independently assigned class labels to avoid the meta-learner using rote memorization to handle tasks [Yin *et al.*, 2020]. The latter aims to alleviate learner overfitting by augmenting the training tasks [Liu *et al.*, 2020]. To simultaneously alleviate these two forms of overfitting, MetaMix [Yao *et al.*, 2021] linearly combines features and labels of samples to increase the mutual-exclusivity and diversity of training tasks. However, these data manipulation strategies are manually designed for specific data or tasks, resulting in a lack of flexibility and generality in real-world applications. For example, most strategies that improve mutual-exclusivity and diversity for classification tasks are difficult to extend to regression tasks [Yin *et al.*, 2020; Jamal and Qi, 2019; Yao *et al.*, 2021]. More importantly, our experiments in Sec. 5.3 show that improving mutual-exclusivity is effective but short-lived, which means that simply increasing task mutual-exclusivity is insufficient to combat memorization overfitting.

In this work, we improve generalization in meta-learning following the gradient regularization perspective and propose a data-independent **M**eta-**G**radient **Aug**mentation (**MGAug**). Considering that the meta-gradient is derived from the inner-loop fine-tuning of base learners, our key idea is to first overcome memorization overfitting in the inner loop by breaking the rote memorization state, and then yield diversity gradients as the meta-gradient augmentation to alleviate learner overfitting in the outer loop. Specifically, memorization break-

*Corresponding author

ing is achieved by pruning the base learner before each inner loop. We explore three different levels of pruning strategies, named *random width pruning (WP)*, *random parameter pruning (PP)*, and *catfish pruning (CP)*[1], respectively. The first two are based on random strategies and inspired by typical GradAug [Yang *et al.*, 2020] and Dropout [Srivastava *et al.*, 2014], respectively, while CP is a newly proposed unstructured pruning strategy that measures a Meta-Memorization Carrying Amount (MMCA) score for each parameter and prunes those with high scores to break the rote state more effectively. Once the rote memorization is removed, the pruned sub-network has to re-fine-tune the remaining parameters to handle new tasks, thus alleviating memorization overfitting. With different pruning rates, sub-networks produce gradients containing diverse task information as a high-quality meta-gradient augmentation to ultimately reduce learner overfitting. Contributions can be summarized in three aspects:

- We propose a novel data-agnostic meta-regularization via meta-gradient augmentation (MGAug), which can alleviate both memorization and learner overfitting in the two-loop meta-learning framework.
- We explore three pruning strategies to break rote memorization, including two existing random prunings and a new *catfish pruning* that measures a Meta-Memorization Carrying Amount (MMCA) score for each parameter.
- We deduce a PAC-Bayes-based generalization bound for two-loop meta-learning with inner-loop pruning and conduct extensive experiments on both mutually-exclusive and non-mutually-exclusive tasks, providing theoretical guarantees and experimental validation.

## 2 Related Work

**Regularization** techniques prevent the model from overfitting the training data and can be roughly divided into *data augmentation*, *label regularization*, and *internal changes*. Data augmentation [Yoo *et al.*, 2020] and label regularization [Li *et al.*, 2020] modify the input and labels, respectively, using various transformations (e.g., flipping and noise addition) to increase sample diversity. In contrast, internal variation [Yang *et al.*, 2020] emphasizes parameter diversity. The well-known Dropout [Srivastava *et al.*, 2014] and its variants randomly remove some neurons to force the learner to capture more features. Shake-Shake [Gastaldi, 2017] and Shake-Drop [Yamada *et al.*, 2019] are designed for a specific residual structure, giving different weights to each residual branch. The network pruning adopted in our MGAug belongs to an internal variation that prunes parameters to break the rote memorization state.

**Meta regularization** is specially designed for meta-learning to solve learner and memorization overfitting [Tabealhojeh *et al.*, 2023]. For learner overfitting, well-designed task augmentation [Liu *et al.*, 2020] remains an effective solution by increasing task diversity. Meta-MaxUp [Ni *et al.*, 2021] splits the meta-framework and further explores various data augmentation combinations. In contrast, memorization overfitting occurs in the inner loop with only a few updates, invalidating most conventional regularization strategies [Yao *et al.*, 2021]. Although constructing mutually-exclusive tasks [Lee *et al.*, 2020] shows promise against memorization overfitting, the task-dependent property makes it difficult to extend mutual-exclusivity to regression and reinforcement learning scenarios [Yin *et al.*, 2020]. MetaPruning [Tian *et al.*, 2020] ignores inner loops and improves meta-generalization through data-agnostic network pruning. Instead, we argue that redundant memories in inner loops are the key cause of memorization overfitting [Guiroy *et al.*, 2019]. MR-MAML [Yin *et al.*, 2020] and TAML [Jamal and Qi, 2019] develop explicit meta-regularization terms to constrain the parameter scale and the base learner behavior, respectively. Unlike them, we directly break the rote memorization fetter via proposed *catfish pruning* and alleviate learner overfitting using derived augmented meta-gradients, which can also be considered an enhanced DropGrad [Tseng *et al.*, 2020].

## 3 Meta Learning

Meta-learning is generally trained and tested on several tasks (here, we omit validation for brevity). To avoid confusion, we use the terms "support set" and "query set" to refer to training and test samples in a single task, leaving "training set" and "testing set" to the meta-learner. Given a set of training tasks $\{\mathcal{T}_t = (D_t^s, D_t^q)\}$ sampled from the task distribution $p(\mathcal{T})$, where $D_t^s = (x_t^s, y_t^s)$ is the support set containing support samples $x_t^s$ and corresponding labels $y_t^s$, and $D_t^q = (x_t^q, y_t^q)$ is the query set containing query samples $x_t^q$ and labels $y_t^q$. The goal of meta-learning is to produce a base learner that can quickly handle new tasks $\mathcal{T}_{new} = (D_{new}^s, D_{new}^q)$, that is, fine-tune the support data $(x_{new}^s, y_{new}^s)$ and then accurately predict $y_{new}^q$ for $x_{new}^q$. Considering the difficulty of constructing a large number of routine tasks, meta-learning algorithms are usually validated on few-shot tasks. When applied to classification scenarios, this is commonly described as a $N$-way $K$-shot task, indicating $K$ samples in the support set, with class labels $y^s$, $y^q \in \{1, \ldots, N\}$.

Most meta-learning methods follow a two-loop framework [Goldblum *et al.*, 2020]. The outer loop first involves sampling a batch of tasks $\{\mathcal{T}_t\}_{t=1}^T \sim p(\mathcal{T})$, and then updating meta-parameters based on the *feedback* derived from the inner loop over these tasks. In each inner loop, with the given meta-parameter $\omega$ and fine-tuning policy $\mathcal{F}$, the base parameters are updated from $\theta(\omega)$ to $\theta_t(\omega)$ on support samples $D_t^s$ for the $t$-th task: $\theta_t(\omega) = \mathcal{F}(\theta(\omega), D_t^s)$. Therefore, the *feedback* is usually defined as the loss on query sample $D_t^q$ derived from the fine-tuned $\theta_t(\omega)$. Let $\mathcal{L}_{outer}$ denote the loss function in the outer loop, and meta-parameters are finally optimized by minimizing empirical risk, i.e., $\arg\min_\omega \sum_t \mathcal{L}_{outer}(\theta_t(\omega), D_t^q)$. Without loss of generality, using the gradient descent, meta-update in the $o$-th outer loop can be further formalized as

$$\omega^o = \omega^{o-1} - \frac{\beta}{T} \sum_{t=1}^{T} \nabla_\omega \mathcal{L}_{outer}\left(\theta_t(\omega^{o-1}), D_t^q\right), \qquad (1)$$

[1] *Catfish pruning* is named after the "catfish effect", which forces sardines to reactivate by putting in catfish to avoid suffocation during transport. Here, sardines are base learner parameters that are forced to fine-tune tasks to avoid rote states by *catfish pruning*.

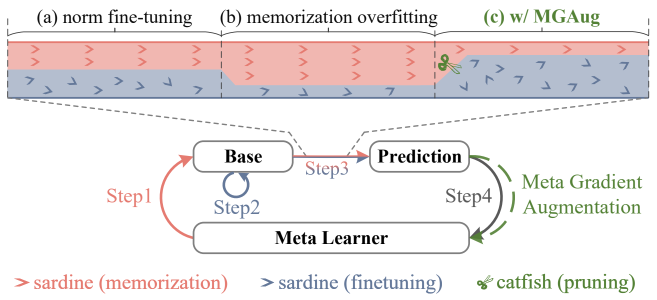


Figure 1: There are four steps in the meta-learner update, including initializing, fine-tuning, predicting, and updating. We zoom in on the inner-loop learning process and present three cases: (a) shows normal base learning, i.e., query samples are jointly predicted by initialization (meta memorization) and fine-tuning. (b) represents memorization overfitting, where query prediction mainly relies on rote memorization. (c) illustrates that the proposed *catfish pruning* breaks meta memorization, forcing the base network to re-predict by fine-tuning and further deriving meta-gradient augmentation.

where $\beta$ is the meta-learning rate and $t$ is the task index.

Note that different meta-parameter deployments $\theta(\omega)$ and fine-tuning algorithms $\mathcal{F}$ motivate different branches of meta-learning methods. The well-known gradient-based meta-learning (GBML) [Lee and Choi, 2018] takes $\omega$ as the initialization of base parameters $\theta$ and fine-tunes them by gradient descent, such as MAML [Finn *et al.*, 2017], Meta-SGD [Li *et al.*, 2017] etc. Formally, let $\mathcal{L}_{inner}$ denote the loss in the inner loop, then $\mathcal{F} \triangleq \arg\min_\theta \mathcal{L}_{inner}(\theta(\omega), D_t^s)$. Similarly, after the $o$-th outer loop, the initialization and the $i$-th update of the base learner are

$$\theta_t^{o,i} = \theta_t^{o,i-1} - \alpha\nabla_\theta\mathcal{L}_{inner}(\theta_t^{o,i-1}, D_t^s), \ldots, \theta_t^{o,0} = \omega^o, \tag{2}$$

where $\alpha$ is the base learning rate. An alternative metric-based meta-learning (MBML) [Snell *et al.*, 2017] meta-learns feature extractors and freezes them in inner loops, i.e., $\theta_t(\omega) = \theta(\omega) = \omega$. This paper focuses on these two meta-learning branches, but MGAug can also be used for other branches [Goldblum *et al.*, 2020] derived from two-loop framework.

## 4 Meta-Gradient Augmentation

This work proposes MGAug to mitigate meta overfitting issues in a data-independent manner. The overall idea is to first overcome memorization overfitting using network pruning and then alleviate learner overfitting with derived augmented meta-gradients. Fig.1 shows the illustration of our MGAug with the proposed *catfish pruning*. Let's start by decomposing the meta-update into four steps: step 1 initializes base parameters $\theta(\omega)$ based on $\omega$; step 2 fine-tunes $\theta(\omega)$ to $\theta_t(\omega)$ on the support set of the $t$-th task; step 3 predicts the query sample based on $\theta_t(\omega)$ and calculates loss values; step 4 updates $\omega$ once based on the average query error over a batch of tasks. Obviously, the query error is the key feedback for the update of meta-parameters and should be inferred jointly by meta-knowledge (i.e., $\omega$) and task-specific fine-tuning (Fig. 1 (a)). Memorization overfitting occurs as $\omega$ is trained enough to directly memorize query predictions while ignoring fine-tuning (Fig. 1 (b)), implying a degradation of rapid adaptability.

To avoid this, MGAug removes the parameters that carry the most meta-memorization using the proposed *catfish pruning* to enforce base learner re-fine-tune to the support set (Fig. 1 (c)). Each pruning is like throwing a catfish into sardines (i.e., base parameters), resulting in different sub-networks and fine-tuning results. The higher the pruning rate, the more significant the memorization break, and the more fine-tuning is required. After several independent pruning and fine-tuning stages, we obtain a set of augmented meta-gradients containing diversity task information, which are ultimately used to update the meta-learner. Taking the GBML as an example, the rest of this section details the inner and outer loops using our MGAug.

### 4.1 Inner-Loop with Network Pruning

Observe the inner loop process after the $o$-th outer loop, where the base parameters are initialized by the latest meta-learned parameters, i.e., $\theta(\omega) \triangleq \theta^{o,0} = \omega^o$. Let $\mathcal{F}_\rho$ denote the pruning criterion. We can obtain the pruned sub-network parameters $\theta_\rho^{o,0} = \mathcal{F}_\rho(\theta^{o,0}, \rho)$ with a given pruning rate $\rho$, and rewrite the $i$-th update on the $t$-th task in (2) as

$$\theta_{\rho,t}^{o,i} = \theta_{\rho,t}^{o,i-1} - \alpha\nabla_\theta\mathcal{L}_{inner}(\theta_{\rho,t}^{o,i-1}, D_t^s). \tag{3}$$

where $\alpha$ is the learning rate in the inner loop. For the pruning criterion $\mathcal{F}_\rho$, we explored three specific strategies whose ability to break rote memorization gradually increases at the same pruning rate. For brevity, we omit the superscripts of the inner and outer loops below.

***Random width pruning (WP)*** is a structural pruning strategy that prunes the neurons in each layer of the network to meet the given pruning rate. Without loss of generality, we use the $l$-$th$ convolutional layer parameter $\theta_{(l)} \in \mathbb{R}^{d_{in}\times d_{out}\times k\times k}$ for illustration, where $k$ represents the convolution kernel size and $d_{in}$ and $d_{out}$ represent the number of input and output channels, respectively. For example, $d_{in}$ is the channel number of input images in the first layer, and $d_{out}$ is the number of corresponding convolution kernels. The number of trainable parameters in this layer is $n_{(l)} = d_{out}\times k\times k$. With the pruning rate $\rho \in [0,1]$, the parameter of the corresponding layer in sub-networks is $\theta_{\rho,(l)}$, where $\left|\theta_{\rho,(l)}\right| = n_{(l)}(1-\rho)$. Inspired by [Yang *et al.*, 2020], we sampled the first $(1-\rho)\times 100\%$ from the entire model as the sub-network, that is, $\theta_{\rho,(l)} \in \mathbb{R}^{d_{in}\times(1-\rho)d_{out}\times k\times k}$.

***Random parameter pruning (PP)*** is an unstructured pruning, where each parameter may be removed individually. Specifically, we introduce an indication mask $m \in \mathbb{R}^n$ consistent with the shape of base parameters $\theta$, where $n$ is the number of parameters and the value of $m$ is randomly selected from the set $\{0,1\}$. A position with a value of 0 in $m$ indicates that the corresponding parameter is pruned, otherwise it is retained. Afterwards, the pruned parameters can be expressed as $\theta_\rho = m\odot\theta$, where $|m| \le n(1-\rho)$ and $\odot$ is the Hadamard product. Compared to WP, PP achieves parameter-level changes, enabling more flexible memorization breaking.

***Catfish pruning (CP)*** is a novel task-oriented pruning strategy that enables stronger memorization breaking based on the current task and parameter state. Like PP, CP also needs an indicator mask $m$. The difference is that the mask value in CP reflects the amount of memory contained in each parameter rather than being randomly generated. Thus, we define the meta-memorization carrying amount and design a memorization-breaking pruning criterion.

**Definition 1.** ***(Meta-Memorization Carrying Amount).*** *Let $\theta \in \mathbb{R}^n$ denote the base learner parameter and $e_{(j)}$ be the indicator vector for the $j$-th parameter $\theta_{(j)}$, whose value is zero everywhere except that index $j$ is one. Keeping everything else constant, we measure the query loss difference in the $t$-th task before and after pruning parameter $\theta_{(j)}$ to get the following Meta-Memorization Carrying Amount (**MMCA**):*

$$\begin{aligned} MMCA_{t,(j)} &\triangleq \Delta\mathcal{L}_{(j)}(\theta; D_t^q) \\ &= \mathcal{L}\left(\mathbf{1} \odot \theta; D_t^q\right) - \mathcal{L}\left(\left(\mathbf{1} - e_{(j)}\right) \odot \theta; D_t^q\right), \end{aligned} \tag{4}$$

*where $\mathbf{1}$ is a vector of all ones with $m$ dimension and $\odot$ denotes the Hadamard product.*

MMCA essentially measures the sensitivity of parameter $\theta_{(j)}$ in solving task $t$. It is reasonable to represent the amount of memorization carried out here since the base parameters are initialized before each inner-loop by the meta parameters derived from the previous outer-loop. However, computing MMCA directly for each discrete parameter is prohibitively expensive as it requires $n+1$ forward passes ($n$ is the number of parameters). By relaxing the binary constraint on the indicator variable, we obtain an approximation of MMCA.

**Proposition 1.** *For any task $\mathcal{T}_t = (D_t^s, D_t^q)$, the change of loss value on $D_t^q$ before and after removing the $j$-th parameter $\theta_{(j)}$ can be approximated by*

$$\Delta\mathcal{L}_{(j)}(\theta; D_t^q) \approx \frac{\partial \mathcal{L}(\theta, D_t^q)}{\partial \theta_{(j)}} \times \theta_{(j)}. \tag{5}$$

We defer the proof to Appendix A. Similar approximation strategies [Koh and Liang, 2017; Lee *et al.*, 2019] have also been used to measure the impact of a data point or connection on the loss. The key difference is that we leverage the fact that the query loss depends on meta-knowledge. Based on this MMCA score, we further compute the value of the binary mask $m$ by the designed memorization-breaking pruning criterion. Similar to PP, the parameters $\theta_{(l)}$ in the $l$-th layer are then pruned by $\theta_{\rho,(l)} = m_{(l)} \odot \theta_{(l)}$, where $||m_{(l)}||_0 / n_{(l)} \le \rho$, where $n_{(l)}$ is the number of parameters in the $l$-th layer.

**Definition 2.** ***(Memorization-breaking pruning criterion).*** *Given a MMCA score mask, the parameters corresponding to the high score positions are removed to break the memorization state as much as possible, i.e.,*

$$m_{t,(j)} = \begin{cases} 0 & \text{if } |MMCA_{t,(j,l)}| \text{ is in the top-}\rho\% \text{ largest value} \\ 1 & \text{otherwise} \end{cases}, \tag{6}$$

*where $(j,l)$ refers to the $j$-th parameter in the $l$-th layer.*

Thus, we obtain a layer-level binary mask and can further compute the sub-network predictions via forwarding propagation. During backward propagation, we apply the same mask to the gradient so that the pruned parameters are no longer updated while the others are trained normally.

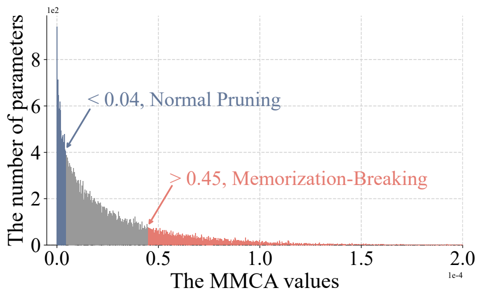


Figure 2: MMCA distribution of the last layer in Conv-4.

**Remark 1.** *CP breaks rote memorization to combat memorization overfitting and generates augmented meta-gradients containing diversity task information, with two differences from connection-sensitivity-based network pruning methods [Koh and Liang, 2017; Tanaka* et al.*, 2020; Frankle* et al.*, 2021]. One is that CP prunes in the inner loop, so the criterion is based on the query set rather than the regular training samples [Wang* et al.*, 2020]. Another essential difference is that normal pruning usually removes insensitive parameters for accuracy preservation, which is even the exact opposite of our memorization-breaking criterion [Frankle and Carbin, 2019; Wang* et al.*, 2022]. We highlight this difference in Fig. 2 by visualizing the MMCA distribution of the last layer parameters in the Conv-4 backbone with a 20% pruning rate.*

## 4.2 Outer-loop with Augmented Meta-Gradients

We now obtain two group gradients with respect to the meta-parameters $w^o$: one is the (original) meta-gradients obtained by backward-propagation on the full network, and the other is derived from several pruned sub-networks. The former retains full meta-knowledge to speed the training process and avoid underfitting at the early learning stage. The latter is the meta-gradient augmentation resulting from network pruning. The meta-parameters are finally updated by accumulating these two-group gradients and formalized as

$$\begin{aligned} \omega^{o+1} = \omega^o - \frac{\beta}{T}\sum_{t=1}^{T}\Big(&\nabla_\omega \mathcal{L}_{outer}(\theta_t(\omega), D_t^q) \\ &+ \sum_{u=1}^{U} \nabla_\omega \mathcal{L}_{outer}(\theta_{u,t}(\omega), D_t^q)\Big), \end{aligned} \tag{7}$$

where $U$ is a hyper-parameter representing the number of sub-networks in each task and $\theta_{u,t}(\omega)$ is the parameter of the $u$-th sub-network for the $t$-th task. Since the meta-learner is trained with shared initialization parameters, it naturally shares diversity attention across all sub-networks, which is the key to combating learner overfitting. The entire procedure of MGAug is provided in Appendix C.

**Remark 2.** *MGAug is designed for the two-loop meta-framework and is quite different from the Dropout-style approaches [Srivastava* et al.*, 2014; Tseng* et al.*, 2020]. The former prunes base parameters before each inner loop in order to break the rote memorization state, while the latter directly prunes meta-parameters in the outer loop, essentially to alleviate learner overfitting.*

**Remark 3.** *The augmented meta-gradients derived from CP in MGAug are based on the task response rather than directly changing training tasks, which is also essentially different from the gradient noise strategy [Tseng* et al.*, 2020].*

## 4.3 A PAC-Bayes Generalization Bound

We provide a PAC-Bayes-based generalization bound [D. A. McAllester, 2013] for the two-loop meta-learning framework with inner-loop pruning, which gives a theoretical guarantee to our MGAug. To simplify the analysis, we prune a sub-network for each task (i.e., $U = 1$). Following the PAC-Bayes meta-learning framework [Amit and Meir, 2018], let $\mathcal{P}$ and $\mathcal{Q}$ be the hyper-prior and hyper-posterior of the meta-learner, and assume that loss function is bounded to the interval $[0, 1]$. For a given pruning rate $\rho \in [0, 1]$ and pruned initial parameters $\omega_t$ of the base learner on task $t$, we take the pruned parameters distribution $Q_{\rho,\omega_t}$ as posterior distribution and the corresponding $Q_{\rho,0} \sim \mathcal{Q}$ as the prior distribution.

**Theorem 1.** ***(Meta-learning PAC-Bayes bound with inner-loop pruning).*** *Let $er(\mathcal{Q})$ and $\hat{er}(Q_{\rho,\omega}, \mathcal{T})$ be the expected and empirical errors in meta-learning, and let $m_t$ denote the number of samples in the $t$-$th$ task. Then for any $\delta \in (0, 1]$ the following inequality holds uniformly for all hyper-posterior distributions $\mathcal{Q}$ with probability at least $1 - \delta$,*

$$\begin{aligned} er(\mathcal{Q}) \leq \frac{1}{T}\sum_{t=1}^{T} \mathop{\mathrm{E}}_{Q_{\rho,0}\sim\mathcal{Q}} \hat{er}_t(Q_{\rho,\omega_t}, \mathcal{T}_t) + \sqrt{\frac{D(\mathcal{Q}\|\mathcal{P}) + \log\frac{2T}{\delta}}{2(T-1)}} \\ + \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{D(\mathcal{Q}\|\mathcal{P}) + \log\frac{2Tm_t}{\delta} + \frac{1-\rho}{2}\|\omega_t\|^2}{2(m_t-1)}}. \end{aligned} \tag{8}$$

The expected error is bounded by the empirical multi-task error plus two complexity terms. The first is the average of task-complexity terms for observed tasks, and the second is the environment-complexity term [Amit and Meir, 2018]. MGAug prunes parameters in the inner loop, reducing the complexity cost of base learners by a factor of $1 - \rho$ and further reducing the task-complexity terms. Appendix B provides proof of Theorem 1.

## 5 Experiments

### 5.1 Experimental Settings

**Datasets.** We conduct experiments with two widely-used datasets: *mini*-ImageNet [Vinyals *et al.*, 2016] and CUB [Wah *et al.*, 2011]. *Mini*-ImageNet consists of 100 classes of natural images, with 600 images per class. It is split into non-overlapping 64, 16, and 20 classes for training, validation, and testing. The CUB contains 200 species of birds and $11,788$ images in total. We randomly select 100 classes as the training set, and the others are equally divided for validation and testing. Experiments involve 5-way 1-shot and 5-shot tasks in both mutually exclusive (ME) and non-mutually-exclusive (NME) settings [Yin *et al.*, 2020].

**Backbones.** We use three backbones with different depths, including Conv-4, ResNet-10, and ResNet-18. The Conv-4 contains four convolution blocks, each block is concatenated by convolution, BatchNorm, nonlinear activation (ReLU), and max pooling layers. The ResNet-10 is a simplified ResNet-18 [He *et al.*, 2016] where only one residual building block is used in each layer. Following previous works [Chen *et al.*, 2019; Yang *et al.*, 2020], we respectively resize images to $84 \times 84$ and $224 \times 224$ before feeding the Conv and ResNet backbones and randomly scale to [84, 64, 48] and [224, 192, 160, 140] as the basic augmentation.

**Baselines.** We choose MAML [Finn *et al.*, 2017] and Prototypical Network [Snell *et al.*, 2017] (abbreviated as ProtoNet) as GBML and MBML instance baselines, respectively. For MAML, we implement a first-order approximation FoMAML for efficiency [Finn *et al.*, 2017]. We further take the transformations designed in Baseline++ [Chen *et al.*, 2019] as the data regularization baseline and mark it with 'Aug'. Below, we mark the pruning strategy with "-XX" and make MGAug-CP as the default setting, abbreviated as MGAug.

**Implementation details.** For 1-shot tasks, we respectively train 4800 and 1600 epochs for GBML and MBML methods, and each epoch includes 100 episodes. For 5-shot tasks, the number of epochs is halved. All results are average results over 600 episodes with confidence intervals of radius one standard error. Following the training procedure of [Chen *et al.*, 2019], all methods are trained from scratch and use the Adam optimizer with $10^{-3}$ learning rate.

### 5.2 Comparison with existing meta-regularization

**MBML-based strategies.** The results of ProtoNet baseline with different meta-regularization methods are listed in Table 1, where the best results are marked in bold and the second-best with an underline. In addition to Aug [Chen *et al.*, 2019], we also compare two state-of-the-art regularization methods designed for the MBML branch, including TaskAug [Liu *et al.*, 2020], Meta-MaxUp [Ni *et al.*, 2021]. Results show that memorization-breaking and augmented diversity gradients greatly improve classification accuracy. For example, in the 5-way 1-shot + ResNet-10 scenario, MGAug improves the accuracy by $6.96\%$ and $6.67\%$ on CUB and *mini*-ImageNet, respectively.

**GBML-based strategies.** Table 2 gives the results of FoMAML baseline with different meta-regularization methods, where the best results are marked in bold and the second with an underline. We compare four state-of-the-art regularization methods designed for the GBML branch, including MR [Yin *et al.*, 2020], TAML [Jamal and Qi, 2019], MetaMix [Yao *et al.*, 2021] and GradDrop [Tseng *et al.*, 2020]. The former two design explicit regularization terms to address memorization and task bias issues in fast adaptation, respectively. The latter two are typical methods of data and gradient regularization, where MetaMix mixes the input and its features using the MixUp strategy and GradDrop randomly drops meta-gradients to increase its diversity. Compared to random-based strategies, gradient diversity in MGAug is learned by different sub-networks on the same task, which leads to self-guided augmentation and higher classification accuracy.

Besides accuracy, we plot the loss and accuracy curves in Fig. 3 to observe meta-overfitting in FoMAML baseline, Aug, and MGAug. Among them, the FoMAML baseline has the lowest loss and the highest accuracy during training, especially in CUB, while it performs the worst over validation epochs. This inversion is powerful evidence of overfitting. In contrast, Aug and MGAug do not significantly overfit the

| | CUB | | | | *mini*ImageNet | | | |
|---|---|---|---|---|---|---|---|---|
| | 5-way 1-shot | | 5-way 5-shot | | 5-way 1-shot | | 5-way 5-shot | |
| | Conv-4 | ResNet-10 | Conv-4 | ResNet-10 | Conv-4 | ResNet-10 | Conv-4 | ResNet-10 |
| ProtoNet | 45.26±0.90 | 50.29±0.89 | 66.25±0.71 | 71.41±0.67 | 31.37±0.62 | 43.54±0.80 | 65.10±0.72 | 63.29±0.67 |
| + Aug | 55.18±0.97 | 71.61±0.87 | 75.93±0.67 | 84.26±0.53 | 44.79±0.82 | 51.65±0.83 | 65.98±0.72 | 74.02±0.65 |
| + TaskAug | 57.64±0.97 | 73.44±0.89 | 78.21±0.65 | 85.78±0.50 | 42.55±0.78 | 56.33±0.89 | 63.97±0.76 | 74.79±0.65 |
| + Meta-MaxUp | 59.79±0.90 | 75.20±0.85 | 78.86±0.57 | 86.02±0.42 | 45.47±0.82 | 57.52±0.82 | 66.22±0.72 | 74.83±0.60 |
| + **MGAug-WP** | 62.35±0.97 | 76.30±0.88 | 79.57±0.62 | 85.19±0.49 | 46.79±0.84 | 57.87±0.86 | 67.31±0.70 | 74.26±0.64 |
| + **MGAug-PP** | 61.28±0.98 | 76.01±0.88 | 79.06±0.61 | 85.74±0.49 | 47.68±0.83 | 55.26±0.86 | 67.84±0.71 | 75.53±0.64 |
| + **MGAug-CP** | **63.00±0.95** | **78.57±0.92** | **80.33±0.62** | **87.09±0.46** | **48.77±0.86** | **58.32±0.86** | **67.99±0.73** | **75.77±0.63** |

Table 1: Classification accuracy with the ProtoNet baseline.

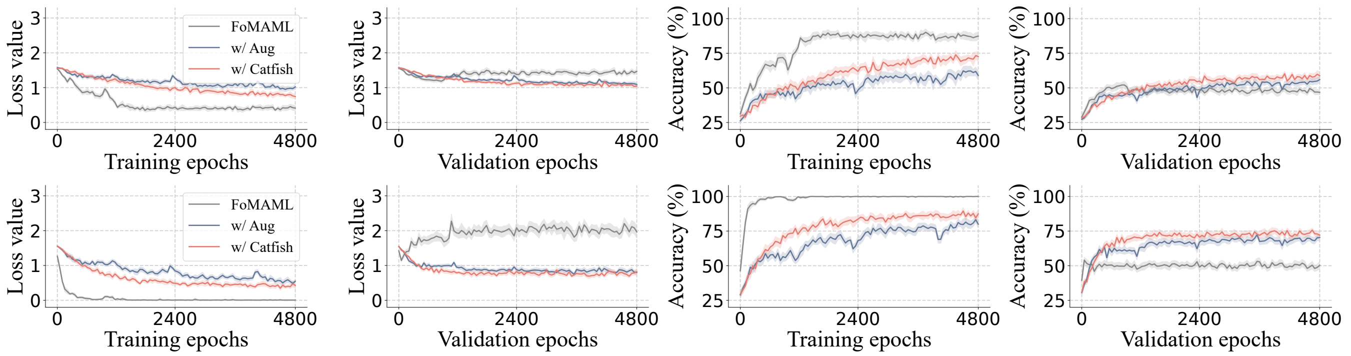


Figure 3: Loss and accuracy curves of FoMAML, Aug, and MGAug using ResNet-10 on *mini*-Imagenet (top) and CUB (bottom). From left to right, the first two are the loss curves during training and validation, respectively, and the last two are the corresponding accuracy curves.

training task and generalize better to unseen tasks. Another interesting trend is the trade-off between training loss and validation accuracy. The training loss of MGAug is always lower than Aug, but it yields more accurate predictions. This phenomenon means that MGAug learns more generalizable meta-knowledge during training.

## 5.3 Behavioral Analysis of MGAug

**Rote memorization breaking.** We observe the behavior of the base learner to investigate the memorization breaking in the inner loop. To this end, we visualize the gap in fine-tuning accuracy between the full-network and sub-networks with different pruning rates via a modified hat graph [Witt, 2019]. Fig. 4 shows the average accuracy of FoMAML with MGAug using ResNet-10 on 100 tasks sampled from CUB. Results for the full-network are indicated by the dashed line with asterisks. The histogram reflects the gap between the accuracy of sub-networks and the full-network, i.e., the upward bar indicates higher accuracy than the full-network and vice versa. The trend in Fig. 4 reflects whether fine-tuning relies on rote memorization or rapid adaptation of meta-knowledge.

i. **NME tasks suffer from severe memorization overfitting.** Observing Fig. 4 (c) and (d), due to label mutual exclusion, each ME task cannot be handled solely on memorization, i.e., the accuracy at step-0 is similar to random classification and improves rapidly after fine-tuning. In contrast, for NME tasks, there is rote memorization in meta-learned parameters that resulted in 28% classification accuracy without fine-tuning (at step-0) and further limited the fine-tuning performance.

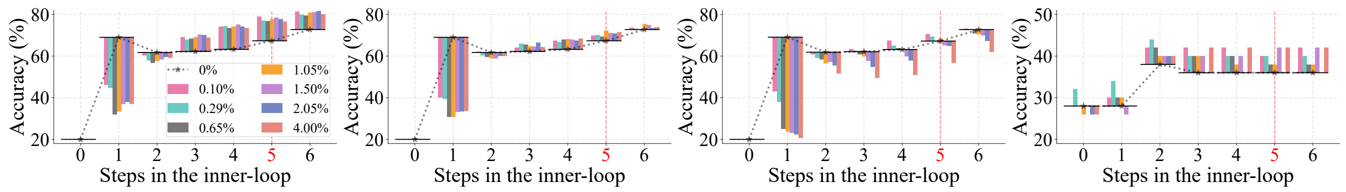
ii. **The ME setting is short-lived for solving memorization issues.** Although the ME setting avoids the reliance on memorization at step-0, the accuracy increases sharply after only one step and remains almost unchanged until the step-5 (fine-tuning five steps by default [Finn *et al.*, 2017]). This trend means that memorization is almost recovered with just one iteration and still prevents subsequent fine-tuning.

iii. **Our MGAug breaks rote memorization.** Unlike constructing ME tasks, MGAug directly breaks memorization and inhibits its recovery. An intuitive phenomenon is that accuracy slowly increases during fine-tuning, even with only a $0.1\%$ pruning rate. Following the same ME setting, Fig. 5 visualizes fine-tuning curves to verify whether fine-tuning is reactivated with broken memorization, which is the key to overcoming memorization overfitting. Further, we plotted the curve fine-tuned from random initialization as a baseline with no memory at all. Clearly, the accuracy of MGAug has a consistent trend with random initialization but improves faster, indicating that the memorization issue is significantly alleviated and fine-tuning is reactivated to rapidly adapt to new tasks.

iv. **CP has stronger breaking capability than WP and PP.** All three prunings hindered memorization recovery, CP was the most effective, followed by PP, and finally WP (see step-1 in Fig. 4 (a), (b), and (c)).

| | CUB | | | | *mini*ImageNet | | | |
|---|---|---|---|---|---|---|---|---|
| | 5-way 1-shot | | 5-way 5-shot | | 5-way 1-shot | | 5-way 5-shot | |
| | Conv-4 | ResNet-10 | Conv-4 | ResNet-10 | Conv-4 | ResNet-10 | Conv-4 | ResNet-10 |
| FoMAML | 53.21±0.48 | 54.69±0.51 | 69.08±0.39 | 64.49±0.43 | 43.50±0.39 | 46.79±0.46 | 59.28±0.38 | 57.86±0.39 |
| + Aug | 53.98±0.48 | 68.14±0.50 | 73.66±0.36 | 78.17±0.36 | 44.41±0.40 | 52.16±0.47 | 60.82±0.38 | 66.93±0.38 |
| + MR | 56.11±0.54 | 69.54±0.52 | 74.73±0.45 | 79.26±0.40 | 44.57±0.41 | 52.65±0.56 | 61.15±0.38 | 68.20±0.51 |
| + TAML | 55.64±0.85 | 70.22±0.81 | 75.39±0.74 | 78.72±0.69 | 45.72±0.30 | 51.78±0.53 | 61.93±0.36 | 67.01±0.77 |
| + GradDrop | 55.39±0.51 | 69.03±0.52 | 74.88±0.38 | 80.59±0.34 | 45.57±0.37 | 52.30±0.47 | 62.35±0.38 | 67.33±0.38 |
| + MetaMix | 57.53±0.53 | 70.38±0.44 | 76.24±0.40 | 80.83±0.35 | **46.06±0.44** | 53.32±0.49 | **62.86±0.38** | 68.81±0.41 |
| + **MGAug-WP** | 56.85±0.50 | 70.73±0.46 | 75.89±0.42 | 81.53±0.44 | 45.65±0.32 | 52.97±0.45 | 61.34±0.40 | 67.53±0.39 |
| + **MGAug-PP** | 55.35±0.47 | 71.17±0.47 | 76.16±0.36 | 81.47±0.35 | 45.77±0.30 | 54.53±0.45 | 62.10±0.41 | 68.95±0.38 |
| + **MGAug-CP** | **58.19±0.49** | **72.14±0.51** | **76.28±0.35** | **81.97±0.33** | 45.95±0.41 | **54.70±0.46** | 62.41±0.40 | **69.27±0.38** |

Table 2: Classification accuracy with the FoMAML baseline.



(a) MGAug-WP in ME tasks (b) MGAug-PP in ME tasks (c) MGAug-CP in ME tasks (d) MGAug-CP in NME tasks

Figure 4: Hat graphs of the base learner's accuracy on ME tasks with (a) WP, (b) PP, and (c) CP. As a comparison to (c), (d) shows the result of MGAug-CP on NME tasks. The dotted line indicates the results of a full network (i.e., 0% pruning), and the histogram indicates the accuracy gap between the full and sub-networks with different pruning rates.

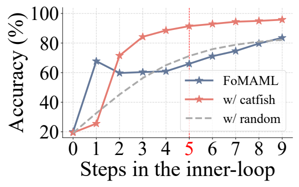


Figure 5: Comparison of the base learner after fine-tuning w/ and w/o *catfish pruning*.

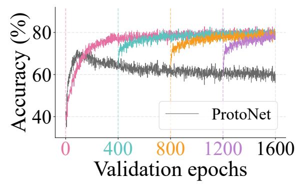


Figure 6: Accuracy trend of training w/ *catfish pruning* starting from different epochs.

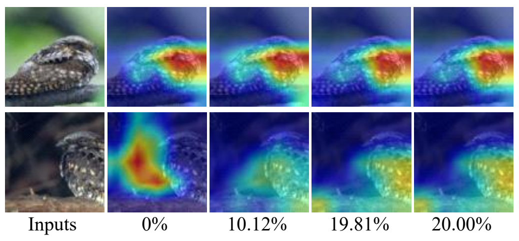


Figure 7: Grad-CAM visualization of two typical CUB samples.

**Augmented meta-gradients.** We empirically infer that the effectiveness of the meta-gradient augmentation derived from pruned sub-networks is twofold. One is the improvement from resolving memorization overfitting, which has been verified in the previous subsection. The other is the diversity of attention introduced by sub-networks with different pruning rates, even for the same task. To verify this, Fig. 7 visualizes the attention regions of different sub-networks using Grad-CAM [Selvaraju *et al.*, 2020] and lists representative examples. Interestingly, attention changes seem to occur more often in samples containing insignificant objects (bottom). Conversely, for the salient ones (top), the learner is more confident in the predictions.

**Plug-and-play property.** In addition, MGAug can also improve meta-generalization in a flexible plug-and-play way. Fig. 6 shows the results of training with MGAug starting from epochs 0, 400, 800, and 1200 on 5-way 5-shot CUB tasks. Both train and test curves show that MGAug consistently improves ProtoNet baseline performance and avoids meta-overfitting, even if it is only used for the last 400 epochs.

## 6 Conclusion

This work proposes a data-independent meta-regularization method, termed MGAug, which alleviates both memorization and learner overfitting in the two-loop meta-learning framework. Unlike existing task augmentation and explicit regularization terms, the key idea is to first solve the rote memorization issue in the inner loop via network pruning and then alleviate learner overfitting with augmented meta-gradients derived from pruned sub-networks. We explore two random pruning strategies and propose a novel *catfish pruning* that achieves the most significant memorization breaking. We deduce a PAC-Bayes-based generalization bound for MGAug. Extensive experimental results show that MGAug significantly outperforms existing meta-learning baselines. We believe that MGAug's ideas can inspire and drive the development of regularization strategies in conventional learning.

## Acknowledgments

The work is supported by Natural Science Foundation of Shandong Province (No. ZR2024MF101, ZR2024MF092), Young Expert of Taishan Scholars (No. tsqn202312026), Shandong Sci-tech SMEs Innovation Project (No. 2024TSGC0740), and Natural Science Foundation of China (No. U23A20389).

## References

[Amit and Meir, 2018] Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended pac-bayes theory. In *ICML*, volume 80, pages 205–214, 2018.

[Chen *et al.*, 2019] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *ICLR*, 2019.

[D. A. McAllester, 2013] D. A. McAllester. A pac-bayesian tutorial with a dropout bound. *CoRR*, abs/1307.2118, 2013.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, volume 70, pages 1126–1135, 2017.

[Frankle and Carbin, 2019] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2019.

[Frankle *et al.*, 2021] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? In *ICLR*, 2021.

[Gastaldi, 2017] Xavier Gastaldi. Shake-shake regularization. *CoRR*, abs/1705.07485, 2017.

[Goldblum *et al.*, 2020] Micah Goldblum, Liam Fowl, and Tom Goldstein. Adversarially robust few-shot learning: A meta-learning approach. In *NeurIPS*, 2020.

[Guiroy *et al.*, 2019] Simon Guiroy, Vikas Verma, and Christopher J. Pal. Towards understanding generalization in gradient-based meta-learning. *CoRR*, abs/1907.07287, 2019.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[Hospedales *et al.*, 2021] Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-learning in neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, PP(99):1–1, 2021.

[Jamal and Qi, 2019] Muhammad Abdullah Jamal and Guo-Jun Qi. Task agnostic meta-learning for few-shot learning. In *CVPR*, pages 11719–11727, 2019.

[Koh and Liang, 2017] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, volume 70, pages 1885–1894, 2017.

[Lee and Choi, 2018] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, volume 80, pages 2933–2942, 2018.

[Lee *et al.*, 2019] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: single-shot network pruning based on connection sensitivity. In *ICLR*, 2019.

[Lee *et al.*, 2020] Haebeom Lee, Taewook Nam, Eunho Yang, and Sung Ju Hwang. Meta dropout: Learning to perturb latent features for generalization. In *ICLR*, 2020.

[Li *et al.*, 2017] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few shot learning. *CoRR*, abs/1707.09835, 2017.

[Li *et al.*, 2020] Weizhi Li, Gautam Dasarathy, and Visar Berisha. Regularization via structural label smoothing. In *AISTATS*, volume 108, pages 1453–1463, 2020.

[Liu *et al.*, 2020] Jialin Liu, Fei Chao, and Chih-Min Lin. Task augmentation by rotating for meta-learning. *CoRR*, abs/2003.00804, 2020.

[Martins *et al.*, 2023] Vinicius Eiji Martins, Alberto Cano, and Sylvio Barbon Junior. Meta-learning for dynamic tuning of active learning on stream classification. *Pattern Recognit.*, 138:109359, 2023.

[Ni *et al.*, 2021] Renkun Ni, Micah Goldblum, Amr Sharaf, Kezhi Kong, and Tom Goldstein. Data augmentation for meta-learning. In *ICML*, volume 139, pages 8152–8161, 2021.

[Rajendran *et al.*, 2020] Janarthanan Rajendran, Alexander Irpan, and Eric Jang. Meta-learning requires meta-augmentation. In *NeurIPS*, 2020.

[Selvaraju *et al.*, 2020] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Int. J. Comput. Vis.*, 128(2):336–359, 2020.

[Snell *et al.*, 2017] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087, 2017.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[Tabealhojeh *et al.*, 2023] Hadi Tabealhojeh, Peyman Adibi, Hossein Karshenas, Soumava Kumar Roy, and Mehrtash Harandi. RMAML: riemannian meta-learning with orthogonality constraints. *Pattern Recognit.*, 140:109563, 2023.

[Tanaka *et al.*, 2020] Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *NeurIPS*, 2020.

[Tian *et al.*, 2020] Hongduan Tian, Bo Liu, Xiao-Tong Yuan, and Qingshan Liu. Meta-learning with network pruning. In *ECCV*, volume 12364, pages 675–700, 2020.

[Tseng *et al.*, 2020] Hung-Yu Tseng, Yi-Wen Chen, Yi-Hsuan Tsai, Sifei Liu, Yen-Yu Lin, and Ming-Hsuan Yang. Regularizing meta-learning via gradient dropout. In *ACCV*, volume 12625, pages 218–234, 2020.

[Vinyals *et al.*, 2016] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, pages 3630–3638, 2016.

[Wah *et al.*, 2011] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[Wang *et al.*, 2020] Chaoqi Wang, Guodong Zhang, and Roger B. Grosse. Picking winning tickets before training by preserving gradient flow. In *ICLR*, 2020.

[Wang *et al.*, 2022] Ren Wang, Haoliang Sun, Xiushan Nie, and Yilong Yin. Snip-fsl: Finding task-specific lottery jackpots for few-shot learning. *Knowl. Based Syst.*, 247:108427, 2022.

[Witt, 2019] Jessica K Witt. Introducing hat graphs. *Cogn. Res. Princ. Implic.*, 4(1):1–17, 2019.

[Xu *et al.*, 2021] Hui Xu, Jiaxing Wang, Hao Li, Deqiang Ouyang, and Jie Shao. Unsupervised meta-learning for few-shot learning. *Pattern Recognit.*, 116:107951, 2021.

[Yamada *et al.*, 2019] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. Shakedrop regularization for deep residual learning. *IEEE Access*, 7:186126–186136, 2019.

[Yang *et al.*, 2020] Taojiannan Yang, Sijie Zhu, and Chen Chen. Gradaug: A new regularization method for deep neural networks. In *NeurIPS*, 2020.

[Yao *et al.*, 2021] Huaxiu Yao, Long-Kai Huang, Linjun Zhang, Ying Wei, Li Tian, James Zou, Junzhou Huang, and Zhenhui Li. Improving generalization in meta-learning via task augmentation. In *ICML*, volume 139, pages 11887–11897, 2021.

[Yin *et al.*, 2020] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. In *ICLR*, 2020.

[Yoo *et al.*, 2020] Jaejun Yoo, Namhyuk Ahn, and Kyung-Ah Sohn. Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy. In *CVPR*, pages 8372–8381, 2020.

[Zhang *et al.*, 2023] Yukun Zhang, Xiansheng Guo, Henry Leung, and Lin Li. Cross-task and cross-domain SAR target recognition: A meta-transfer learning approach. *Pattern Recognit.*, 138:109402, 2023.